Program Synthesis with Large Language Models

Year: 2021 | Citations: 2733 | Authors: Jacob Austin, Augustus Odena, Maxwell Nye

Abstract

This paper explores the limits of the current generation of large language models for program synthesis in general purpose programming languages. We evaluate a collection of such models (with between 244M and 137B parameters) on two new benchmarks, MBPP and MathQA-Python, in both the few-shot and fine-tuning regimes. Our benchmarks are designed to measure the ability of these models to synthesize short Python programs from natural language descriptions. The Mostly Basic Programming Problems (MBPP) dataset contains 974 programming tasks, designed to be solvable by entry-level programmers. The MathQA-Python dataset, a Python version of the MathQA benchmark, contains 23914 problems that evaluate the ability of the models to synthesize code from more complex text. On both datasets, we find that synthesis performance scales log-linearly with model size. Our largest models, even without finetuning on a code dataset, can synthesize solutions to 59.6 percent of the problems from MBPP using few-shot learning with a well-designed prompt. Fine-tuning on a held-out portion of the dataset improves performance by about 10 percentage points across most model sizes. On the MathQA-Python dataset, the largest fine-tuned model achieves 83.8 percent accuracy. Going further, we study the model's ability to engage in dialog about code, incorporating human feedback to improve its solutions. We find that natural language feedback from a human halves the error rate compared to the model's initial prediction. Additionally, we conduct an error analysis to shed light on where these models fall short and what types of programs are most difficult to generate. Finally, we explore the semantic grounding of these models by fine-tuning them to predict the results of program execution. We find that even our best models are generally unable to predict the output of a program given a specific input.